WWW.SCIAM-MAGAZINE.NAROD.RU

Как оперировать с тысячезначными числами и зачем это нужно

ФРЕД ГРЮЕНБЕРГЕР

От редактора. Статью под рубрикой «Занимательный компьютер» для этого номера журнала подготовил профессор Грюенбергер — специалист по кибернетике и вычислительной математике из Калифорнийского университета в Нортридже. Вычислительными машинами Грюенбергер заинтересовался более 40 лет назад; с тех пор он опубликовал 28 книг по различным вопросам в этой области, а с 1973 по 1981 г. был главным редактором ежемесячного журнала "Popular Computing".

Начиная со следующего номера рубрику «Занимательный компьютер» будет вести А. К. Дьюдни — доцент факультета кибернетики и вычислительной математики Университета в Западном Онтарио. Научные интересы Дьюдни в основном касаются теории вычислений, но многим читателям он больше известен как исследователь в области двумерных «пространств», которым была посвящена статья М. Гарднера в июльском номере журнала "Scientific American" за 1980 г. Собственные взгляды Дьюдни на эту проблему изложены в его недавно выпущенной издательством Poseidon Press книге "The Planiverse: Computer Contact with a Two-dimensional World". В следующем году издательство Computer Science Press выпустит в свет сборник статей Дьюдни под названием "The Turing Omnibus", который будет посвящен различным проблемам кибернетики и вычислительной математики.

ЕСЛИ НА вашем калькуляторе есть клавиша для возведения чисел в квадрат, то попробуйте проделать следующее: наберите число 1.0000001, а затем 27 раз нажмите эту клавишу. То, что вы проделали, эквивалентно возведению набранного числа в степень 134217728. Правильным результатом с точностью до 10-й значащей цифры будет 674 530.4707, но ваш калькулятор почти наверняка даст другой ответ. Эта задача специально предназначена для демонстрации того, какую точность вычислений может дать ваш калькулятор. В таблице на с. 112 приведены результаты, полученные на нескольких калькуляторах и на ЭВМ по программам, написанным на версиях Бейсика и Фортрана. Ни одно из этих технических средств не дало даже семи верных цифр.

На большинстве электронных калькуляторов операция возведения в квадрат не равнозначна введению числа и умножению его на то же самое число. В операции умножения размер (или точность) сомножителей ограничивается числом разрядов, которые могут быть изображены на табло вычислительного устройства. А процедура возведения в квадрат осуществляется с помощью такого представления числа, хранящегося в вычислительном устройстве, которое, как правило, включает несколько дополнительных «контрольных разрядов», и они каждый раз участвуют в процессе вычисления, но оператор их не видит. Так, если вы извлекаете квадратный корень из 2 на калькуляторе с восьмиразрядным табло и двумя контрольными разрядами, то на табло высветится число 1.4142136, но внутри устройства хранится число 1.414213562. Если вы теперь нажмете клавишу возведения в квадрат, то на табло появится 2.0000000, в то время как при умножении числа 1.4142136 на себя получим 2.000000106.

При вычислениях ошибка в седьмой значащей цифре (разряде) в большинстве случаев несущественна. Допустим, однако, что вычисления производятся на ЭВМ по программе, в которой предусмотрено выполнение одной последовательности команд, если число точно равно 2, и другой последовательности, если оно хоть на сколько-нибудь отличается от 2; в этом случае неточность вычислений может привести к печальным последствиям. Для того чтобы не рисковать, можно использовать, пожалуй, самый надежный прием — округлить полученное значение до определенной степени точности, прежде чем проверять его на равенство 2. Бывают случаи, когда способ округления не помогает. При решении задач, связанных с многократным последовательным возведением в квадрат дробной части, единственная возможность улучшить результат — это постоянно поддерживать все бóльшую точность на всех стадиях вычислительного процесса.

Для многих задач, связанных с расчетом, вполне достаточно точности до восьмого или девятого знака, как, например, при операциях с чековыми книжками, если только сумма не превышает 1 млн. долл. Ни одна из физических постоянных не известна с точностью более 12 значащих цифр. Однако, для того чтобы обеспечить даже умеренную точность конечного результата, в процессе вычислений может потребоваться значительно более высо-

ЧИСЛО ВОЗВЕДЕНИЙ В КВАДРАТ	ПОКАЗАТЕЛЬ СТЕПЕНИ	РЕЗУЛЬТАТ ПРИ ТОЧНОСТИ ДО 8-ГО РАЗРЯДА	РЕЗУЛЬТАТ ПРИ ТОЧНОСТИ ДО 15-ГО РАЗРЯДА
0	1	1.0000001	1.00000010000000
1	2	1.0000002	1.00000020000001
2	4	1.0000004	1.00000040000006
3	8	1.0000008	1.00000080000028
4	16	1.0000016	1.00000160000120
5	32	1.0000032	1.00000320000496
6	64	1.0000064	1.00000640002016
7	128	1.0000128	1.00001280008128
8	256	1.0000256	1.00002560032640
9	512	1.0000512	1.00005120130818
10	1024	1.0001024	1.00010240523794
11	2048	1.0002048	1.00020482096271
12	4096	1.0004096	1.00040968387705
13	8192	1.0008192	1.00081953559497
14	16384	1.0016391	1.00163974282853
15	32768	1.0032809	1.00328217441361
16	65536	1.0065726	1.00657512149610
17	131072	1.0131884	1.01319347521490
18	262144	1.0265507	1.02656101821804
19	524288	1.0538063	1.05382752412486
20	1048576	1.1105077	1.11055245060312
21	2097152	1.2332274	1.23332674554061
22	4194304	1.5208498	1.52109486126578
23	8388608	2.3129841	2.31372957696917
24	16777216	5.3498954	5.35334455534193
25	33554432	28.621381	28.6582979282091
26	67108864	819.18345	821.298040141993
27	134217728	671061.52	674530.470741078

Результаты последовательного возведения в квадрат числа 1.0000001 при ограниченной точности вычислений

WWW.SCIAM-MAGAZINE.NAROD.RU

кая точность. Когда число 1.0000001 возводится в квадрат 27 раз подряд, то для получения 10 точных значащих цифр промежуточные вычисления должны вестись с 15-значными числами.

ПРИМЕРОМ задачи, в которой обеспечение высокой точности вычислений является непременным условием, может служить непрерывный поиск все бóльших простых чисел. До появления электронных цифровых вычислительных машин в течение многих лет не было известно простого числа, большего $2^{127} - 1$. В десятичной форме оно записывается 39 десятичными цифрами. Начиная с 1952 г. с помощью ЭВМ было поставлено 16 новых рекордов; в настоящее время наибольшее из известных простых чисел — это $2^{132049} - 1$, состоящее из 39 751 знака. Чтобы проверить, является ли число простым, т.е. установить, что оно делится только на 1 и само на себя, все необходимые для этого арифметические операции должны производиться с исключительно высокой точностью.

Вычисление значения числа π — еще одна давняя задача, требующая исключительно высокой точности. На протяжении многих лет рекорд удерживал английский математик Уильям Шенкс: производя вычисления вручную, он получил 528 правильных знаков (и еще 179, которые оказались неверными). Сейчас π известно с точностью до 8 388 608-го знака.

Учитывая, что механизация выполнения арифметических действий с высокой точностью началась более 30 лет назад и что стоимость вычислений на ЭВМ с тех пор постоянно снижается, а их доступность растет, можно подумать, что перечень известных результатов стал довольно обширным. На самом же деле чисел, которые известны с высокой точностью, не так уже много. Приведем некоторые из них.

Квадратный корень из 2 известен до одного миллиона десятичных знаков, а кубический корень из 16 — до 1000 знаков.

Значение действительного корня уравнения Уоллиса $X^3 - 2X - 5 = 0$ известно до 4000 знаков. (Это уравнение английский ученый XVII в. Дж. Уоллис выбрал для иллюстрации численного решения уравнений методом Ньютона; с тех пор оно служит средством проверки многих других методов аппроксимации.)

Г. Нелсон (который когда-то вместе с Д. Словинским удерживал рекорд в получении максимального значения из известных простых чисел), сотрудник Ливерморской национальной лаборатории им. Лоуренса, вычислил 1 000 000!, т.е. произведение $1\,000\,000 \times 999\,999 \times \times \ldots \times 2 \times 1$. Результат записывается числом из 5 565 709 цифр, и его распечатка на ЭВМ представляет собой пачку бумаги толщиной более 12 см.

```
1.0000001

1.00000020000001

1.000000400000060000040000001

1.000000800000280000056000007000000560000028000000800000001

1.000001600001200000560000182000043680008008001144000128700011440000800800
0436800018200000560000012000000160000001

1.000003200004960004960003596002013760906192336585705183028048806451225290
2450257928747373647143565657227801080446572276714356347373622579285290244 8
6451224280488010518300336585609061920201376003596000049600000496000003200 0
0001

1.000006400020160041664063537676245194974430121663461681220599659328917561
1024560167423597795591072087154275016633545248630060887595709345161126819 9
1538318569142373529010725957109077232359858708556491166159373567974201161
4343435727721880857335556894375111059951058069637780618096826744935403780 3
9908345463896857916128955102413621530314804648290127250139140645477741557 96
9713217570058495461654301216199497436876245120635376004166400020160000064 0
000001

1.000012800081280341377066802645669423620652593817045586724716639752509356
3706604966887868632339808586665878774084866169288885995073744765475334516 8
0029637400737469957105560072416700993009294401078544080725549112523177899 2
8499808013093965927574941133244381547633598449089802869082118094468776017 8
2665118692282090047510399909402800454116374069753973388842836767559325442 3
0479144789181357521024292881433616534591710408142167764017463463439708597 2
8390095407066078152415625359663834513241403709468936478003065366991757932 0
3926349554600257842099023918809283454906896251044486185892911937500254243 4
5723208897691093624362353259851586247224033739625021480333630107329307139 2
4088117757151012106967826289242967898516694703304135759892818613724712834 4
6374823047835854926003256454000295386933291445449058223011356249373289045 8
5715015608707021749702661852579574236112264566401066800003413760008128000 0
1280000001
```

Точные результаты многократного возведения в квадрат числа 1.0000001

Для задачи, известной как «проблема 196», потребовалось 50 тыс. вычислительных шагов, и на каждом из них приходилось оперировать числами, имеющими 21 000 знаков. Другой «орешек» — «проблема 3N + 1» была исследована вплоть до 1000-значных чисел. Более подробно о той и другой задачах мы поговорим ниже.

В последовательности Фибоначчи (1,1,2,3,5,8,13,21 . . . , где каждое последующее число есть сумма двух предыдущих) точно вычислены первые 10 000 членов. Последние числа этого ряда имеют более 2000 знаков.

Число Эйлера e, являющееся основанием натурального логарифма, вычислено с точностью более чем до 125 000 знаков.

Можно привести еще несколько примеров. Так, Р. Госпер, сотрудник фирмы Symbolics, Inc., используя метод, основанный на оперировании с непрерывными дробями, вычислил 2800 знаков корня седьмой степени из 306. (Результат не имеет особого значения, и эта задача была выбрана случайно исключительно для проверки правильности метода.)

БОЛЬШИНСТВО из приведенных выше примеров взято из той категории задач, в решении которых высокая точность вычислений играет определяющую роль; задачи эти примечательны, в сущности, только тем, что при их решении нужно оперировать числами значительно более «длинными», чем нам представляется необходимым. Следующая задача, требующая высокой точности вычислений, более распространена, имеет практический смысл и потому менее искусственна. Возникает она при нахождении корней квадратного уравнения вида $AX^2 + BX + C = 0$. Значение выражения $B^2 - 4AC$, называемого дискриминантом уравнения, определяет, являются корни действительными или комплексными. Если дискриминант положительный, корни действительные, а если отрицательный — то комплексные; если же $B^2 - 4AC$ точно равно нулю, уравнение имеет два одинаковых действительных корня. Отсюда следует, что даже незначительная ошибка в вычислении дискриминанта может привести к качественно иному результату при решении уравнения.

Подобная чувствительность к малым ошибкам в вычислении может проявиться и в решении системы однородных линейных уравнений. Рассмотрим систему

$$53.17X - 18.91Y - 5.67Z = -174.65,$$
$$-12.65X + 36.16Y - 47.08Z = 298.59,$$
$$303.80X - 203.03Y + 112.89Z = -1769.02.$$

WWW.SCIAM-MAGAZINE.NAROD.RU

КАЛЬКУЛЯТОРЫ	РЕЗУЛЬТАТ	ОШИБКА (%)
Texas Instruments SR-52	674520.6053	0.00146
Hewlett-Packard 33, 67, 41C	674494.0561	0.00540
Sharp Electronics EL506	674492.75	0.00559
Monroe Calculator 1930	674383.1672	0.02183
Texas Instruments 30	674363.69	0.02473
ЯЗЫКИ ПРОГРАММИРОВАНИЯ И КОМПЬЮТЕРЫ		
Фортан двойной точности (CDC Cyber)	674530.5363	0.00000973
Восьмиразрядный фортран (CDC Cyber)	674530.5765	0.00001568
Бейсик для Apple II	22723.9709	96.63114
Бейсик для персональных компьютеров IBM	8850273.	1212.06423
Бейсик для Ontel	8886690.	1217.46401

Точность, обеспечиваемая некоторыми машинами и языками программирования при многократном возведении в квадрат

Система составлена таким образом, что всем трем уравнениям удовлетворяют значения $X = -3$, $Y = 2$ и $Z = -4$, но это не единственное ее решение. В самом деле, хотя это и не очевидно с первого взгляда, система не может иметь единственного решения, поскольку два из ее уравнений описывают параллельные плоскости. (Третье уравнение представляет собой разность между первым уравнением, умноженным на 5, и умноженным на 3 вторым уравнением.)

Чтобы убедиться в том, что единственного решения нет, нужно вычислить так называемый определитель матрицы коэффициентов. Определитель выражается суммой всех возможных произведений элементов, которые не принадлежат ни одному и тому же столбцу, ни одной и той же строке. Если определитель окажется равным нулю, то система уравнений не имеет решения и в этом случае дальнейшие вычисления не нужны (или, что более важно, в этом месте программы следует предусмотреть останов). В данном случае, однако, определитель вычислялся по схеме, называемой корзиночным плетением, и при точности вычислений до девятого десятичного знака был получен результат, отличный от нуля, а именно — 0.000202179. В данном примере можно было бы получить правильный ответ с помощью других методов, но в иных случаях и это бы не помогло. Дело в том, что при точности до девятого знака нельзя получить надежный результат. Кроме того, приведенная система уравнений — небольшая, и ее коэффициенты не имеют более шести значащих цифр; для более сложных систем проблема становится острее.

193	194	195	196	197	198	199	200
391	491	591	691	791	891	991	002
1124	685	786	887	988	1089	1190	202
4211	586	687	788	889	9801	0911	
5335	1271	1473	1675	1877	10890	2101	
	1721	3741	5761	7781	09801	1012	
	2992	5214	7436	9658	20691	3113	
		4125	6347	8569	19602		
		9339	13783	18227	40293		
			38731	72281	39204		
			52514	90508	79497		
			41525	80509			
			94309	171017			
			90349	710171			
			187088	881188			
			880781				
			1067869				
			9687601				
			10755470				
			07455701				
			18211171				
			17111281				
			35322452				
			25422353				
			60744805				
			50844706				
			111589511				
			⋮				

Вычисления по правилам «проблемы 196» для нескольких начальных чисел

ПРИ РЕШЕНИИ многих задач теории чисел и других разделов математики требуется исключительно высокая точность представления очень больших целых чисел. Примером может служить упомянутая «проблема 196». Возьмем любое положительное число — двух- или более значное. Переставим цифры в обратном порядке и вновь полученное число сложим с исходным. Снова поменяем порядок цифр в сумме на обратный и сложим полученное число с предыдущим. Будем делать так до тех пор, пока не придем к палиндрому — числу, прямой и обратный порядок цифр в котором одинаков (см. рис. слева). В большинстве случаев палиндром получается быстро; если, например, начать с числа 195, то уже через четыре шага получим палиндром. Наименьшее число, для которого с помощью описанной процедуры палиндром, как представляется, никогда не получится, — это 196: как уже отмечалось, начиная с числа 196, было проделано 50 тыс. шагов и они не дали результата. Среди первых 100 000 целых чисел имеется 5996 чисел, которые, по-видимому, не приводят к палиндрому независимо от того, как долго мы будем продолжать вычисления (хотя это предположение и не доказано).

Проблема $3N + 1$ рассматривалась в мартовском номере нашего журнала. Начнем с любого положительного числа N; если оно нечетное, заменим его на $3N + 1$, а если четное, поделим N пополам. Продолжим эту процедуру до тех пор, пока не получим 1. Например, если начать с $N = 9$, то, последовательно применяя указанное правило, получим: 9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2 и 1 — всего 20 чисел. Эта незамысловатая задача содержит немало загадок. Всегда ли полу-

WWW.SCIAM-MAGAZINE.NAROD.RU

чаемая в процессе решения последовательность заканчивается 1 независимо от исходного значения N? Имеется ли какой-либо способ определения необходимого числа шагов, т.е. можно ли составить такую формулу, которая позволяла бы по значению N предсказывать число шагов? Всегда ли по любому заданному числу шагов можно определить некое нечетное N, для которого потребуется сделать именно столько шагов? (Четное N по любому заданному числу шагов найти легко: это 2, возведенное в степень, на 1 меньшую заданного числа шагов.)

В 1980 г. я высказал предположение, что среднее количество целых чисел, получаемых в процессе вычислений в задаче $3N + 1$, приблизительно определяется выражением $24.64\ D - 101$, где D — количество цифр исходного числа N. Эта оценочная формула была получена на основе вычислений, выполненных почти до 200-значных чисел, которые тогда представлялись мне довольно большими. Используя программу, обеспечивающую высокую точность вычислений, я имел возможность проверить свое допущение на нескольких бóльших числах. Для 1000-значного числа по моей формуле получается, что окончательный результат достигается через 25 539 шагов. Я обнаружил, что когда N = 1. . . (998 нулей). . . 1, или $10^{1000} + 1$, то ряд достигает 1 через 23 069 шагов. Число 55. . . (997 нулей). . . 1 порождает последовательность из 24 413 чисел. Следовательно, мы получили некое подтверждение того, что средняя скорость сходимости устойчива и предсказуема.

На малых значениях N была замечена тенденция (которая, по-видимому, должна была сохраниться и для больших чисел) порождать ряды одинаковой длины многими последовательными значениями N. Действительно, можно заметить, как такие ряды чисел появляются все чаще по мере увеличения N. Например, все 230 последовательных целых чисел, начиная с числа 912345678912345678900000000000000000000001, порождают ряды из 997 членов. (Заметим, что наша эмпирическая формула для этого числа дает 959 членов.)

Точность арифметических операций, которые непосредственно могут быть выполнены центральным процессором обычного микрокомпьютера, весьма ограниченная. Во многих вычислительных машинах единственными операциями, выполняемыми в явном виде, являются сложение и вычитание целых чисел, «длина» которых не превышает 16 битов, или двоичных разрядов: самое большое число в десятичной системе счисления, которым может оперировать такая ЭВМ, — это 65 536, так что уровень точности не превышает пяти десятичных знаков. Даже на самых мощных компьютерах точность выше нескольких десятков десятичных цифр может быть обеспечена только такой программой, в которой содержится группа элементарных операций, позволяющих разбить большое число на более мелкие части.

На с. 114 приведена блок-схема алгоритма вычислений с высокой точностью — процедуры, по которой было получено наибольшее из известных простых чисел: $2^{132049} - 1$. Программа ориентирована на определенный микропроцессор — модель 6502 фирмы MOS Technology. Этим микропроцессором комплектуются компьютеры, выпускаемые фирмами Apple Computer, Inc., Commodore Business Machines, Inc. и др. Алгоритм можно легко приспособить и к другим микропроцессорам.

Самой большой сложностью, с которой приходится сталкиваться при разработке такой программы, является, пожалуй, выбор способа представления чисел в памяти машины. Процессор может оперировать только двоичными числами, а нам желательно получать результат вычислений в десятичной форме. Приемлемым компромиссом является использование двоично-десятичной системы счисления, в которой каждая десятичная цифра представляется в соответствующей ей двоичной записи. Микропроцессор 6502 организует машинную память в «байтах» — последовательностях из восьми двоичных разрядов (битов), и таким образом обеспечивается удобный способ хранения десятичного числа, при котором каждая его цифра занимает в памяти по одному байту. (Нельзя сказать, чтобы этот способ был наиболее эффективным, но зато он самый простой.)

Первый шаг в алгоритме — очистить машинную память емкостью 39 760 байтов. Для этого все байты заполняются нулями. На следующем шаге исходная величина, равная 1, записывается в очищенную рабочую область, так что теперь в памяти записана последовательность из 39 759 нулей, завершающаяся единицей. В счетчик, который потребуется для соблюдения последовательности вычислительных операций, в самом начале засылается нуль.

Главной частью программы является цикл, в котором происходит периодическое обращение к подпрограмме удвоения числа, хранимого в рабочей области памяти. После каждого такого удвоения к содержимому счетчика прибавляется единица и полученное число сравнивается с 132 049. Если этот предел еще не достигнут, вновь вызывается подпрограмма удвоения; если же в счетчике накопилось предельное число, программа выходит из цикла. На последнем этапе из значения, хранящегося в рабочей области памяти, вычитается единица и выводится результат.

На блок-схеме программы, изображенной на с. 114 слева, подпрограмма удвоения выделена цветом, а набор составляющих ее команд представлен блок-схемой справа. Всякий раз, когда производится обращение к подпрограмме, начальный адрес рабочей области памяти, где хранится самая правая, или младшая, цифра исследуемого числа, присваивается индексу (X). Содержимое этого адреса затем удваивается путем прибавления к нему такого же числа. Так же как и при сложении вручную, основная трудность здесь состоит в необходимости «переносить» в процессе суммирования единицу с одного разряда на другой. Если предыдущее сложение выработало разряд переноса, его необходимо добавить к новому результату. Последний в свою очередь должен быть проверен на последующий перенос: если сумма больше 9, из нее вычитается 10 и она заменяется на остаток, а в следующий разряд переносится единица. Этот процесс повторяется для всех 39 760 байтов рабочей области машинной памяти.

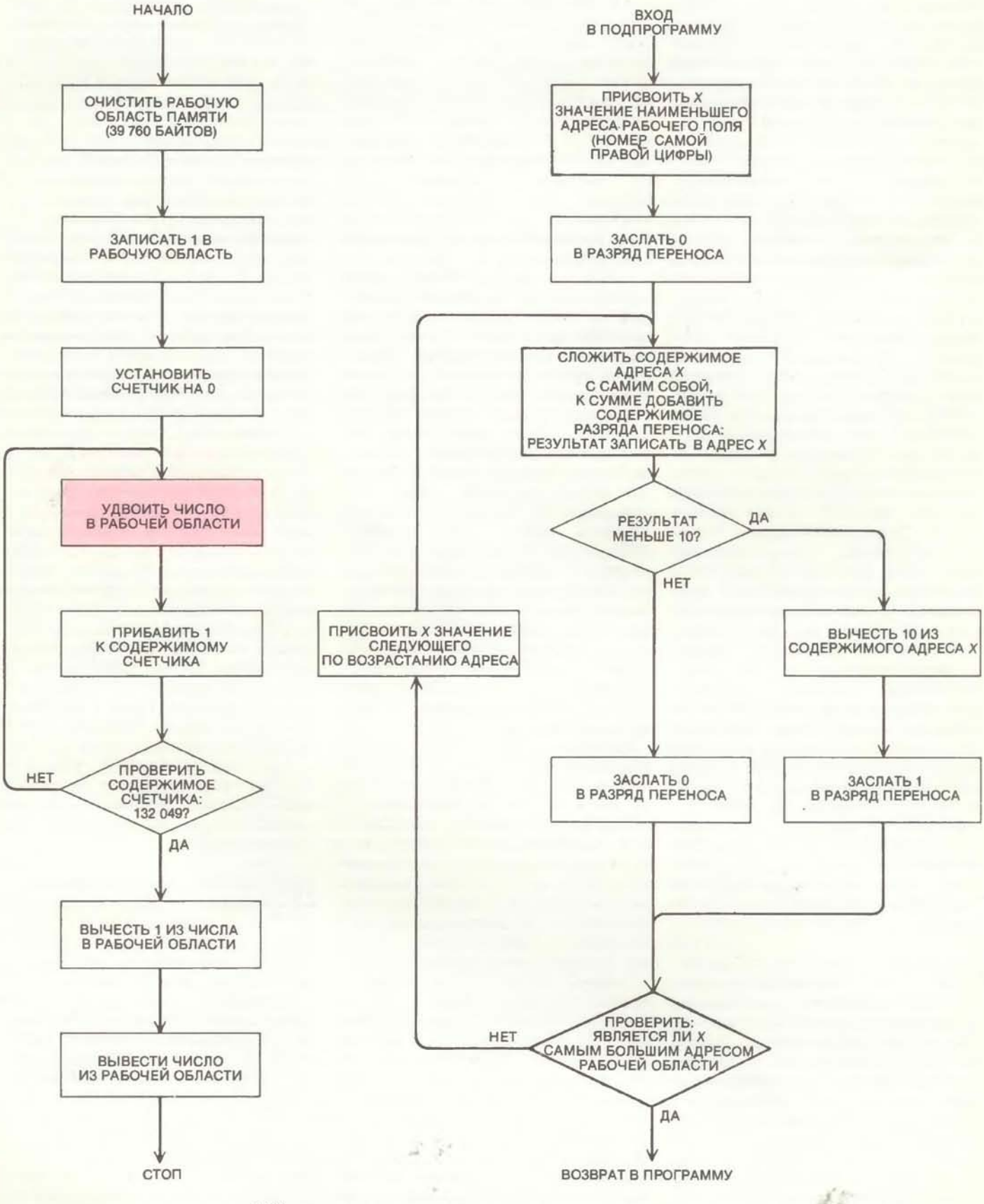
Процессор 6502 компьютера модели Apple II способен выполнять более 250 тыс. операций в секунду. И все же на выполнение программы, представленной на с. 114, потребовалось 120 ч машинного времени, чтобы подсчитать 132 049-ю степень 2. Это время можно значительно уменьшить за счет простого «усечения» вычислительного процесса. Например, на ранних стадиях вычислений нет необходимости удваивать все цифры, записанные в памяти машины, так как большинство из них нули. Рабочую область можно считать заполненной, когда в ней будет находиться, скажем, 150 байтов, и это число будет увеличиваться на три байта при каждом увеличении степени на 10. При таком подходе в программу придется ввести дополнительные команды, но зато быстродействие при этом возрастает.

Все числа, которыми приходится оперировать при вычислении $2^{132049} - 1$,— целые, но существует немало задач, решение которых облегчается, если использовать такую же запись, которая принята в научных работах, когда число представляется в виде произведения десятичной дроби, называемой мантиссой, и степени 10 (показатель степени в такой форме записи называется порядком). Например, текущий год можно записать в виде 1.984×10^3, где 1.984 — мантисса, а 3 — порядок. Программы высокой точности, которые оперируют числами в такой форме, сложнее, чем аналогичные программы, рассчитанные на действия только с целыми числами, по-

WWW.SCIAM-MAGAZINE.NAROD.RU

тому что в первом случае каждое число имеет несколько составляющих (помимо мантиссы и порядка в него входят знак мантиссы и знак порядка).

В большинстве языков программирования высокого уровня предусмотрены некоторые средства, позволяющие использовать такую форму представления чисел; в этом случае говорят о системе с плавающей точкой. Числа меньше определенного размера (количества разрядов) представляются на табло как обычная десятичная дробь, но внутри машины эти числа хранятся в виде мантиссы и порядка. Поле, отведенное для различных элементов числа, определяет точность и диапазон значений этого числа. Отводя больше



Вычисление простого числа $2^{132049} - 1$, состоящего из 39 751 цифры; блок-схемы программы (слева) и подпрограммы (справа)

WWW.SCIAM-MAGAZINE.NAROD.RU

места для мантиссы, мы повышаем точность, а чем больше порядок, тем шире диапазон чисел. Версия языка Бейсик в программном обеспечении для Apple II имеет фиксированную точность — примерно 9 десятичных знаков.

Для обеспечения еще более высокой точности необходимо опять прибегнуть к возможностям программного обеспечения. Г. Робинсон, будучи сотрудником Ливерморской лаборатории, составил пакет программ вычислений с высокой точностью для чисел, представленных с плавающей точкой. Программы, написанные на машинном языке микропроцессора 6502, могут обеспечивать любой уровень точности вплоть до 600 десятичных знаков и допускают величину порядка до 9999. Эти предельные параметры были выбраны, исходя из соответствующих характеристик процессора: 600-значная мантисса, 4-значный порядок и их знаки можно записать 256 байтами, а в модели 6502 блок емкостью 256 байтов — это одна «страница» памяти.

В программах Робинсона операции, которые могут выполняться над числами, включают основные арифметические действия, логарифмирование, возведение в степень, извлечение квадратного корня, а также вычисление различных тригонометрических функций. Предусмотрено вычисление и не столь широко распространенных функций, как, например, функции Эйлера и преобразования Вейнгартена для суммирования медленно сходящихся рядов. В пакете имеются значения 26 постоянных и 8000 простых чисел. Пакетом можно пользоваться как весьма совершенным калькулятором или обращаться к отдельным его программам как к подпрограммам. Первые версии показали, что арифметические действия и вычисления функций могут производиться с предельной точностью до 600 десятичных знаков.

Для тех, кто проявляет интерес к численным процедурам, первое знакомство с цифровым компьютером иногда приводит в уныние; это потому, что элементарные вычисления на машине выполняются быстро и, что самое важное, безупречно, и как следствие возникает ложное представление, будто тщательный численный анализ можно выполнить без всяких усилий. После этого трудно поверить, что корень пятой степени из 100 (константа, которая используется для сравнения величины звезд) нельзя вычислить намного точнее, чем с помощью ручного калькулятора. Пакет программ, подобный пакету Робинсона, позволяет оправдать возлагаемую на компьютер надежду. Корень пятой степени из 100 можно вычислить до 100 десятичных знаков буквально за несколько минут.

Библиография

АЛМАЗНЫЕ НАКОВАЛЬНИ В ФИЗИКЕ ВЫСОКИХ ДАВЛЕНИЙ

THE DIAMOND CELL AND THE NATURE OF THE EARTH'S MANTLE. William A. Bassett in *Annual Review of Earth and Planetary Sciences*, Vol. 7, pages 357—384; 1979.

INTERIORS OF THE GIANT PLANETS. D. J. Stevenson in *Annual Review of Earth and Planetary Sciences*, Vol. 10, pages 257—295; 1982.

DIAMOND ANVIL CELL AND HIGH-PRESSURE PHYSICAL INVESTIGATIONS. A. Jayaraman in *Reviews of Modern Physics*, Vol. 55, No. 1, pages 65—108; January, 1983.

Стишов С. М. СОВРЕМЕННОЕ СОСТОЯНИЕ ФИЗИКИ ВЫСОКИХ ДАВЛЕНИЙ. — Вестник АН СССР, 1981, № 9, с. 52—65.

Жарков В. Н. ВНУТРЕННЕЕ СТРОЕНИЕ ЗЕМЛИ И ПЛАНЕТ. — М.: Наука, 1983.

МОЛЕКУЛЯРНЫЕ ОБЛАКА, ЗВЕЗДООБРАЗОВАНИЕ И СТРОЕНИЕ ГАЛАКТИК

THE NEWEST STARS IN ORION. Gareth Wynn-Williams in *Scientific American*, Vol. 245, No. 2, pages 46—55; August, 1981.

MOLECULAR CLOUDS IN GALAXIES. Mark Morris and L. J. Rickard in *Annual Review of Astronomy and Astrophysics*, Vol. 20, pages 517—545; 1982.

THE SEARCH FOR INFRARED PROTOSTARS. C. G. Wynn-Williams in *Annual Review of Astronomy and Astrophysics*, Vol. 20, pages 587—618; 1982.

GIANT MOLECULAR-CLOUD COMPLEXES IN THE GALAXY. Leo Blitz in *Scientific American*, Vol. 246, No. 4, pages 84—94; April, 1982.

THE MOLECULAR GAS DISTRIBUTION IN M51. Nick Scoville and Judith S. Young in *The Astrophysical Journal*, Vol. 265, No. 1, Part 1, pages 148—165; February 1, 1983.

Центр Галактики. Под ред. Г. Риглера и Р. Блендфорда. Перев. с англ. — М.: Мир, 1984.

СМЕШАННЫЕ ВЫЧИСЛЕНИЯ

Ершов А. П. О СУЩНОСТИ ТРАНСЛЯЦИИ. — Программирование, 1977, № 5, с. 21—39.

Ершов А. П. СМЕШАННЫЕ ВЫЧИСЛЕНИЯ: ПОТЕНЦИАЛЬНЫЕ ПРИМЕНЕНИЯ И ПРОБЛЕМЫ ИССЛЕДОВАНИЯ. В кн.: Методы математической логики в проблемах искусственного интеллекта и систематическое программирование. Ч. 2. — Вильнюс: Институт математики и кибернетики АН Литовской ССР, 1980, с. 26—55.

INCREMENTAL COMPUTATION. Lionello Lombardi in *Advances in Computers*, edited by F. L. Alt and M. Rubinoff. Academic Press, 1967, Vol. 8, pages 248—334.

PARTIAL COMPUTATION OF PROGRAMS. Yoshihiko Futamura in RIMS *Symposia on Software Science and Engineering*, Kyoto 1982 Proceedings, edited by Eiiti Goto et al. Lecture Notes in *Computer Science*. Springer-Verlag, 1982, Vol. 147, pages 1—35.

ТОРНАДО

SEVERE LOCAL STORMS. Edited by David Atlas. *Meteorological Monographs*, Vol. 5, No. 27; 1963.

SEVERE THUNDERSTORM EVOLUTION AND MESOCYCLONE STRUCTURE AS RELATED TO TORNADOGENESIS. Leslie R. Lemon and Charles A. Doswell III in *Monthly Weather Reviews*, Vol. 107, No. 9, pages 1184—1197; September, 1979.

TORNADO DYNAMICS. R. P. Davies-Jones in *Thunderstorms: A Social, Scientific, and Technical Documentary*; *Vol. 2. Thunderstorm Morphology and Dynamics*, edited by E. Kessler. U. S. Government Printing Office, 1982.

A REVIEW OF RECENT ADVANCES IN TORNADO VORTEX DYNAMICS. John T. Snow in *Reviews of Geophysics and Space Physics*, Vol. 20, No. 4, pages 953—964; November, 1982.

Наливкин Д. В. УРАГАНЫ, БУРИ, СМЕРЧИ. — Л.: Наука, 1969.

Прох Л. З. СМЕРЧИ И ШКВАЛЫ. — М.: Знание, 1981.

ЗРЕНИЕ ЧЕЛОВЕКА И ТЕХНИЧЕСКИЕ СИСТЕМЫ ВИДЕНИЯ

ARTIFICIAL INTELLIGENCE. Patrick Henry Winston. Addison-Wesley Publishing Co., 1977.

A COMPUTATIONAL THEORY OF HUMAN STEREO VISION. D. Marr and T. Poggio in *Proceedings of the Royal Society of London*, B, Vol. 204, No. 1156, pages 301—328; May 23, 1979.

FROM IMAGES TO SURFACES: A COMPUTATIONAL STUDY OF THE HUMAN EARLY VISUAL SYSTEM. William Eric Leifur Grimson. The MIT Press, 1981.

THEORETICAL APPROACHES IN NEUROBIOLOGY. Edited by Werner E. Reichardt and Tomaso Poggio. The MIT Press, 1981.

INTENSITY, VISIBLE-SURFACE, AND VOLUMETRIC REPRESENTATIONS. H. K. Nishihara in *Artificial Intelligence*, Vol. 17, Nos. 1—3, pages 265—284; August, 1981.

VISION. David Marr. W. H. Freeman and Company, 1982.